A Novel Framework for Online Supervised Learning with Feature Selection

Lizhe Sun, Adrian Barbu

Florida State University

abarbu@stat.fsu.edu

January 9, 2021

Online Learning

- **Online Learning**
  1. In big data learning, we often encounter datasets so large that they cannot fit in the computer memory.
  2. Online learning methods are capable of addressing these issues by constructing the model sequentially, one example at a time.
  3. We assume that the samples are i.i.d / adversary.

The Framework for An Online Learning Algorithm

- Assuming $\mathbf{w}_1 = 0$, and we only can access data samples $\{(\mathbf{x}_i, y_i) : i = 1, 2, \cdots\}$ streaming in one at a time.
- **for** i = 1, 2 $\cdots$

  Receive observation $\mathbf{x}_i \in \mathbb{R}^n$

  Predict $\hat{y}_i$

  Receive the true value $y_i \in \mathbb{R}$

  Suffer the loss function $f(\mathbf{w}_i; \mathbf{z}_i)$ in which $\mathbf{z}_i = (\mathbf{x}_i, y_i)$

  Update $\mathbf{w}_{i+1}$ from $\mathbf{w}_i$ and $\mathbf{z}_i$
- **end**
- **Target**: minimize the cumulative loss $\frac{1}{n}\sum_{i=1}^{n} f(\mathbf{w}_i; \mathbf{z}_i)$.

Literature Review: Stochastic Gradient Descent (SGD)

- Stochastic Gradient Descent (SGD) is the most widely used traditional online learning algorithm.
- The original idea can be traced back to Robbins and Monro (1951) and Kiefer et al. (1952).
- However, the SGD algorithm cannot select features.

Literature Review: Online Learning with Sparsity

- To learn a better model, we need to consider feature selection in online learning.
- Langford et al. (2009) proposed the framework of truncated gradient.
- Shalev-Shwartz and Tewari (2011) designed stochastic mirror descent.
- Truncated second order methods in Fan et al. (2018); Langford et al. (2009); Wu et al. (2017).

Literature Review: OPG and RDA

- Two main frameworks for online learning with regularization
  1. Online Proximal Gradient Descent (OPG)
  2. Regularized Dual Averaging (RDA)
- OPG is designed by Duchi and Singer (2009) and Duchi et al. (2010), and RDA is proposed by Xiao (2010).
- Some variants, designed by Suzuki (2013) and Ouyang et al. (2013).
  1. OPG-SADMM
  2. RDA-SADMM

Literature Review

Hazan et al. (2007)

- An online Newton method
- Uses a similar idea with running averages, to update the inverse of the Hessian matrix
- Has $\mathcal{O}(p^2)$ computational complexity
- Did not address the issues of variable standardization and feature selection.

Literature Review: Summary

- The classical online learning algorithms, such as SGD, cannot select features.
- In recent years, many new online learning algorithms are proposed to select features.
- However, no matter in theory or numerical experiments, the proposed algorithms cannot recover the true features. This concern motivates us to develop our running averages framework.

Framework of Running Averages Algorithms

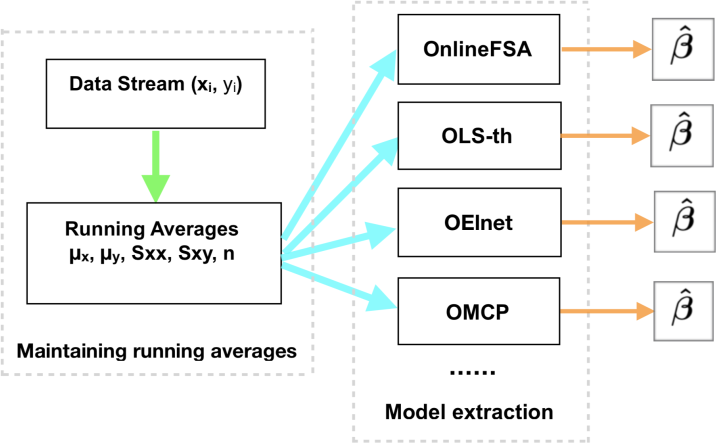

Figure: The running averages are updated as the data is received. The model is extracted from the running averages only when desired.

Running Averages

We have samples $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^T \in \mathbb{R}^p$ and responses $\mathbf{y}_i \in \mathbb{R}$, we can compute running averages as follows:

- $\mathbf{S}_x = \boldsymbol{\mu}_x = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i, \mathbf{S}_y = \mu_y = \frac{1}{n}\sum_{i=1}^n y_i$
- $\mathbf{S}_{xx} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T$
- $\mathbf{S}_{xy} = \frac{1}{n}\sum_{i=1}^n y_i\mathbf{x}_i$
- $\mathbf{S}_{yy} = \frac{1}{n}\sum_{i=1}^n y_i^2$
- Sample size: n

Can be updated online, e.g.

$$\boldsymbol{\mu}_x^{(n+1)} = \frac{n}{n+1}\boldsymbol{\mu}_x^{(n)} + \frac{1}{n+1}\mathbf{x}_{n+1}.$$

Standardization of Running Averages

- Standardization is important for feature selection
- Let $\mathbf{D} = \mathrm{diag}(1/\,\mathrm{std}(\mathbf{X}_1), ..., 1/\,\mathrm{std}(\mathbf{X}_p))$
- The standardization of data matrix $\mathbf{X}$ and vector $\mathbf{y}$
  - $\tilde{\mathbf{X}} = (\mathbf{X} - \mathbf{1}_n \boldsymbol{\mu}_x^T)\mathbf{D}$
  - $\tilde{\mathbf{y}} = (\mathbf{y} - \mu_y \mathbf{1}_n)$
- The equivalent standardization using running averages:
  - $\mathbf{S}_{\tilde{x}\tilde{y}} = \frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{y}} = \frac{1}{n}\mathbf{D}\mathbf{X}^T\mathbf{y} - \mu_y \mathbf{D}\boldsymbol{\mu}_x = \mathbf{D}\mathbf{S}_{xy} - \mu_y \mathbf{D}\boldsymbol{\mu}_x$
  - $\mathbf{S}_{\tilde{x}\tilde{x}} = \frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} = \mathbf{D}(\frac{\mathbf{X}^T\mathbf{X}}{n} - \boldsymbol{\mu}_x\boldsymbol{\mu}_x^T)\mathbf{D} = \mathbf{D}(\mathbf{S}_{xx} - \boldsymbol{\mu}_x\boldsymbol{\mu}_x^T)\mathbf{D}$
- We will assume data is standardized in all algorithms below

Online Least Squares (**OLS**)

- Normal equations

$$\frac{1}{n}\mathbf{X}^T\mathbf{X}\beta = \frac{1}{n}\mathbf{X}^T\mathbf{y}.$$

- Since $\frac{1}{n}\mathbf{X}^T\mathbf{X}$ and $\frac{1}{n}\mathbf{X}^T\mathbf{y}$ are running averages, we obtain:

$$\mathbf{S}_{xx}\beta = \mathbf{S}_{xy}.$$

- Thus, online least squares is equivalent to offline least squares.

Online Least Squares with Thresholding (**OLSth**)

- Aimed at solving the following constrained minimization problem:

$$\min_{\boldsymbol{\beta}, \|\boldsymbol{\beta}\|_0 \le k} \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2.$$

- A non-convex problem because of the sparsity constraint

Algorithm 1 Online Least Squares with Thresholding (OLSth)

Input: Running averages $\mathbf{S}_{xx}, \mathbf{S}_{xy}$, sample size n, sparsity level k.

Output: Trained regression parameter vector $\boldsymbol{\beta}$ with $\|\boldsymbol{\beta}\|_0 \le k$.

1: Fit the model by OLS, obtaining $\hat{\boldsymbol{\beta}}$

2: Keep only the k variables with largest $|\hat{\beta}_j|$

3: Fit the model on the selected features by OLS

Online Feature Selection with Annealing (**OFSA**)

- An iterative thresholding algorithm (Barbu et al., 2017).
- Simultaneously estimates coefficients and selects features.
- Uses the gradient $\frac{\partial}{\partial \boldsymbol{\beta}} \frac{\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2}{N} = \mathbf{S}_{xx}\beta - \mathbf{S}_{xy}$, which can be updated online.
- Uses an annealing schedule M_e to gradually remove features

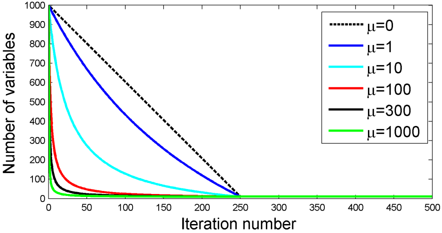

Figure: Different annealing schedules M_e vs iteration number e.

Algorithm 2 Online Feature Selection with Annealing (OFSA)

Input: Running averages $\mathbf{S}_{xx}, \mathbf{S}_{xy}$, sample size n, sparsity level k, annealing parameter μ.

Output: Trained regression parameter vector $\boldsymbol{\beta}$ with $\|\boldsymbol{\beta}\|_0 \leq k$.

Initialize $\boldsymbol{\beta} = 0$.

for $t = 1$ to N^{iter} **do**

  Update $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta(\mathbf{S}_{xx}\boldsymbol{\beta} - \mathbf{S}_{xy})$

  Keep only the M_t variables with highest $|\beta_j|$ and renumber them $1, ..., M_t$.

end for

Fit the model on the selected features by OLS.

Online Lasso and Online Adaptive Lasso

- The Lasso estimator, proposed in (Tibshirani, 1996), solves the optimization problem

$$\arg\min_{\boldsymbol{\beta}} \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|,$$

where $\lambda > 0$ is a tuning parameter.

- However, because Lasso estimator cannot recover the true features, Zou (2006) proposed the adaptive Lasso, which solves the weighted Lasso

$$\arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_n \sum_{j=1}^{p} w_j|\beta_j|, j = 1, 2, \cdots, p,$$

where w_j is the weight for β_j. We can use the OLS coefficients as weights when $n > p$.

Algorithm 3 Online Adaptive Lasso (OALa)

Input: Running averages $\mathbf{S}_{xx}, \mathbf{S}_{xy}$, sample size n, penalty λ.

Output: Trained sparse regression parameter vector $\boldsymbol{\beta}$.

1: Compute the OLS estimate $\hat{\boldsymbol{\beta}}^{ols}$.

2: Let $\boldsymbol{\Sigma}_w = \mathrm{diag}|\hat{\boldsymbol{\beta}}^{ols}|$.

3: Denote $\mathbf{S}^w_{xx} = \boldsymbol{\Sigma}_w \mathbf{S}_{xx} \boldsymbol{\Sigma}_w$ and $\mathbf{S}^w_{xy} = \boldsymbol{\Sigma}_w \mathbf{S}_{xy}$

Initialize $\boldsymbol{\beta} = 0$.

for $t = 1$ to N^{iter} **do**

  Update $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta(\mathbf{S}^w_{xx}\boldsymbol{\beta} - \mathbf{S}^w_{xy})$

  Update $\boldsymbol{\beta} \leftarrow S_{\eta\lambda}(\boldsymbol{\beta})$ ($S_{\eta\lambda}(\cdot)$ is the soft thresholding operator).

end for

Fit the model on the selected features by OLS.

Model Selection

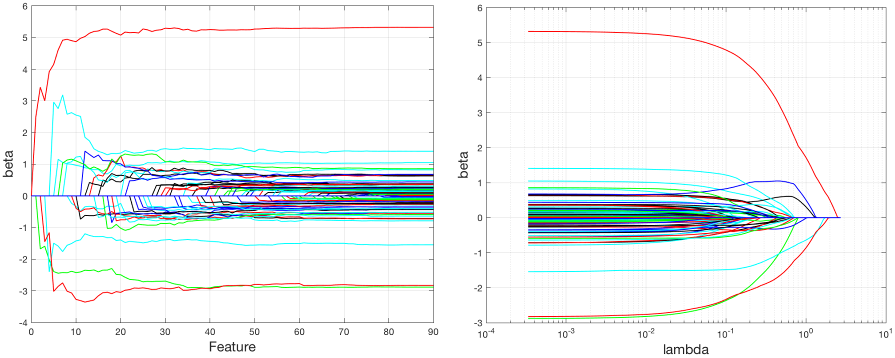

Figure: The solution path for online OLSth (Left) and online Lasso (Right) for the Year Prediction MSD dataset.

Online MCP

- We also can cover non-convex penalties into our running averages framework, such as MCP (Zhang, 2010).
- The MCP solves

$$\arg\min_{\boldsymbol{\beta}} \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \mathbf{P}(\boldsymbol{\beta}, \lambda), \text{ where}$$

$$\mathbf{P}(\boldsymbol{\beta}, \lambda) = \lambda \sum_{j=1}^{p} \text{sign}(\beta_j) \int_0^{|\beta_j|} (1 - \frac{x}{\lambda b})_+ \mathrm{d}x,$$

where $b > 0$ is a fixed parameter.

- Zhang (2010) proved that MCP can recover the support of the true features with high probability.

First, we define the MCP thresholding operator:

$$\boldsymbol{\Theta}_{\mathrm{MCP}}(t;\lambda)=\begin{cases}0 & \text{if } 0\le|t|\le\lambda,\\ \frac{t-\mathrm{sign}(t)\lambda}{1-1/b} & \text{if } \lambda<|t|\le b\lambda,\\ t & \text{if } |t|>b\lambda.\end{cases}$$

Algorithm 4 Online MCP (OMCP)

Input: Running averages $\mathbf{S}_{xx}, \mathbf{S}_{xy}$, sample size n, penalty λ.
Output: Trained sparse regression parameter vector $\boldsymbol{\beta}$.

Initialize $\boldsymbol{\beta}=0$.
for $t=1$ to N^{iter} **do**
  Update $\boldsymbol{\beta}\leftarrow\boldsymbol{\beta}-\eta(\mathbf{S}_{xx}\boldsymbol{\beta}-\mathbf{S}_{xy})$
  Update $\boldsymbol{\beta}\leftarrow\boldsymbol{\Theta}_{\mathrm{MCP}}(\boldsymbol{\beta};\eta\lambda)$
end for
Fit the model on the selected features by OLS.

Online Classification by Running Averages

- Unlike regression, we cannot use running averages to design classification algorithms directly.
- But we can use the methods above and apply them as is for classification problems
- There are theoretical guarantees for true feature recovery for some of them

The equivalence between online algorithms by running averages and offline algorithms

Proposition

Consider the penalized regression problem

$$\min_{\boldsymbol{\beta}} \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \mathbf{P}(\boldsymbol{\beta}; \lambda),$$

where $\mathbf{P}(\boldsymbol{\beta}; \lambda) = \sum_{j=1}^{p} \mathbf{P}(\beta_j; \lambda)$ *is a penalty function. It is equivalent to the online optimization based on running averages:*

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\boldsymbol{\beta}^T \mathbf{S}_{xx}\boldsymbol{\beta} - \boldsymbol{\beta}^T \mathbf{S}_{xy} + \mathbf{P}(\boldsymbol{\beta}; \lambda).$$

True Feature Recovery for OLSth

Proposition

Suppose we have the linear model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon},\ \boldsymbol{\epsilon} \sim N(0, \sigma^2\mathbf{I}),$$

where $\mathbf{X} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_n^T]^T$ *is the data matrix, with* $\mathbf{x}_i \in \mathbb{R}^p$, $i = \overline{1, n}$. *Let* $S_{\boldsymbol{\beta}^*} = \{j, \beta_j^* \neq 0\}$, $|S_{\boldsymbol{\beta}^*}| = k^*$ *and*

$$\min_{j \in S_{\boldsymbol{\beta}^*}} |\beta_j^*| > \frac{4\sigma}{\sqrt{\lambda}}\sqrt{\frac{\log(p)}{n}}, 0 < \lambda \leq \lambda_{\min}(\frac{1}{n}\mathbf{X}^T\mathbf{X}).$$

Then with probability $1 - 2p^{-1}$, *the index set of top* k^* *values of* $|\hat{\beta}_j|$ *is exactly* $S_{\boldsymbol{\beta}^*}$.

Regret

- In the theoretical analysis of online learning, it is of interest to bound the regret:

$$R_n = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{w}_i; \mathbf{z}_i) - \min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{w}; \mathbf{z}_i),$$

which measures what is lost compared to offline learning, in a way measuring the convergence speed of online algorithms.

Regret Analysis

Theorem

(Regret of OLSth) *With some mild assumptions for* $\mathbf{X}_{S_{\beta^*}}$*, if the true* β^* *satisfies*

$$\min_{j \in S_{\boldsymbol{\beta}^*}} |\beta_j^*| > \frac{4\sigma}{\lambda}\sqrt{\frac{\log(p)}{\sqrt{n}}}, \text{ for } \sqrt{\lambda} < 0.9\lambda_{\min}(\sqrt{\boldsymbol{\Sigma}}) - \sqrt{\frac{p}{n_0}}. \quad (1)$$

where $n_0 = \max(p+1, 400\log(n), \frac{1}{4}\left(\frac{2\log(n)}{\log(p)}+1\right)^2) > p$*, then with probability at least* $1 - 3/n$ *the regret of OLSth satisfies:*

$$R_n = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\boldsymbol{\beta}_i)^2 - \min_{\|\boldsymbol{\beta}\|_0 \le k}\frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i^T\boldsymbol{\beta})^2 \le \mathcal{O}(\frac{\log^2(n)}{n}).$$

Classification

Theorem

(True support recovery) *Consider the special case of a single index model,* $\mathbf{y} = G\{h(\mathbf{X}\beta^*) + \epsilon\}$*, in which* $\mathbf{X} \sim \mathcal{N}(0, \mathbf{\Sigma})$ *and* $\mathbf{\Sigma}$ *satisfies the irrepresentable condition. If* G*,* h *are known strictly increasing continuous functions and under the assumptions from Neykov et al. (2016), the least squares Lasso algorithm can recover the support of true features correctly for discrete response* $\mathbf{y}$*.*

Memory and Computational Complexity

- In general, the memory complexity for the running averages is $\mathcal{O}(p^2)$ because S_{xx} is a $p \times p$ matrix.
- The computational complexity of maintaining the running averages is $\mathcal{O}(np^2)$.
- Except OLSth, the computational complexity for obtaining the model using the running average-based algorithms is $\mathcal{O}(p^2)$ based on the limited number of iterations, each taking $\mathcal{O}(p^2)$ time.
- As for OLSth, it is at most $\mathcal{O}(p^3)$ because we need to solve a system.

Simulated Data Experiments

- Data with uniformly correlated predictors: given a scalar α, we generate $z_i \sim \mathcal{N}(0,1)$, then we set

$$\mathbf{x}_i = \alpha z_i \mathbf{1}_{p\times 1} + \mathbf{u}_i, \text{ with } \mathbf{u}_i \sim \mathcal{N}(0, \mathbf{I}_p).$$

  Finally we obtain the data matrix $\mathbf{X} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \cdots, \mathbf{x}_N^T]^T$.
- Correlation between any two variables is $\alpha^2/(1+\alpha^2)$, and we set $\alpha = 1$ in our experiments.
- Given $\mathbf{X}$, the dependent response $\mathbf{y}$ is generated from the linear regression model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\varepsilon}, \text{ with } \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I}_n).$$

  where $\boldsymbol{\beta}^*$ is a p-dimensional sparse parameter vector.
- The true coefficients $\beta_j^* = 0$ except $\beta_{10j^*}^* = \beta$, $j^* = 1, 2, \cdots, k$, where β is signal strength value.

Numerical Results - Regression

	Variable Detection Rate (%)									Test RMSE								
n	Lasso	SGD	SIHT	SADMM	OLSth	OFSA	OALa	OElnet	OMCP	Lasso	SGD	SIHT	SADMM	OLSth	OFSA	OALa	OElnet	OMCP
$p = 1000, k = 100$, strong signal $\beta = 1$																		
10^3	32.14	-	11.22	18.10	77.40	**99.96**	81.05	32.12	91.27	11.63	9.424	23.15	95.05	5.592	**1.072**	5.045	11.61	3.405
$3 \cdot 10^3$	46.05	-	11.22	41.23	**100**	**100**	**100**	45.19	99.93	9.464	8.772	13.45	93.50	**1.017**	**1.017**	**1.017**	9.557	1.047
10^4	72.40	-	11.22	65.78	**100**	**100**	**100**	72.42	**100**	6.07	7.913	13.34	94.92	**1.003**	**1.003**	**1.003**	6.042	**1.003**
$p = 1000, k = 100$, weak signal $\beta = 0.1$																		
10^3	31.33	-	10.89	17.53	11.92	**77.64**	13.15	31.33	69.98	1.557	1.387	2.522	9.560	1.728	**1.197**	1.712	1.555	1.244
$3 \cdot 10^3$	44.85	-	10.89	40.11	95.57	**98.68**	95.77	44.11	95.17	1.389	1.335	1.674	9.392	1.044	**1.024**	1.042	1.403	1.044
10^4	70.53	-	10.89	62.48	**100**	**100**	**100**	71.10	**100**	1.183	1.276	1.663	9.541	**1.003**	**1.003**	**1.003**	1.176	**1.003**
$p = 1000, k = 100$, weak signal $\beta = 0.01$																		
10^3	**14.09**	-	10.89	13.53	10.11	12.15	11.34	14.08	13.53	1.128	**1.022**	1.027	1.363	1.069	1.201	1.060	1.124	1.128
10^4	**31.58**	-	10.89	19.80	22.48	26.64	23.16	31.54	32.52	1.009	1.007	1.007	1.370	1.025	1.021	1.024	**1.006**	1.005
10^5	81.93	-	10.89	11.30	80.55	**85.19**	80.84	81.80	85.03	**1.001**	1.005	1.010	1.382	1.003	1.003	1.003	1.003	1.003
$3 \cdot 10^5$	98.66	-	10.89	10.80	98.94	**99.28**	98.96	98.71	99.27	0.999	1.002	1.008	1.383	**0.998**	**0.998**	**0.998**	**0.998**	**0.998**
10^6	-	-	10.89	-	**100**	**100**	**100**	**100**	**100**	-	0.997	1.005	-	**0.996**	**0.996**	**0.996**	**0.996**	**0.996**
$p = 10000, k = 1000$, strong signal $\beta = 1$																		
10^4	22.80	-	10.20	24.01	98.09	**99.56**	98.80	22.76	41.71	40.05	29.38	42.21	913.4	4.606	**2.415**	3.675	40.72	33.48
$3 \cdot 10^4$	26.64	-	10.20	10.22	**100**	**100**	**100**	26.48	69.38	37.11	27.82	42.01	924.6	**1.017**	**1.017**	**1.017**	36.99	20.58
10^5	-	-	10.20	8.89	**100**	**100**	**100**	34.65	95.48	-	24.73	41.75	860.8	**1.006**	**1.006**	**1.006**	33.35	6.972
$p = 10000, k = 1000$, weak signal $\beta = 0.1$																		
10^4	22.69	-	10.22	21.03	14.51	**98.64**	14.9	22.91	41.63	4.219	3.097	4.326	92.51	4.351	**1.128**	4.337	4.194	3.502
$3 \cdot 10^4$	26.69	-	10.22	8.76	**100**	**100**	**100**	26.46	68.84	3.819	2.957	4.321	93.51	**1.017**	**1.017**	**1.017**	3.838	2.314
10^5	-	-	10.22	8.87	**100**	**100**	**100**	34.60	95.25	-	2.666	4.291	86.09	**1.006**	**1.006**	**1.006**	3.485	1.230
$p = 10000, k = 1000$, weak signal $\beta = 0.01$																		
10^4	21.89	-	10.21	17.03	10.07	**31.23**	10.48	21.83	26.92	1.113	1.058	1.089	9.118	1.144	**1.076**	1.143	1.105	1.090
$3 \cdot 10^4$	25.87	-	10.21	9.30	35.02	**52.45**	35.14	26.12	43.86	1.070	**1.043**	1.086	9.228	1.108	1.046	1.108	1.079	1.056
10^5	-	-	10.21	10.19	77.32	**83.78**	77.35	33.37	74.11	-	1.035	1.083	8.368	1.025	**1.016**	1.024	1.061	1.022
$3 \cdot 10^5$	-	-	10.21	9.92	98.53	**98.96**	98.53	45.66	96.08	-	1.026	1.082	7.482	1.002	**1.001**	1.002	1.043	1.003
10^6	-	-	10.21	-	**100**	**100**	**100**	72.54	**100**	-	1.009	1.079	-	**1.000**	**1.000**	**1.000**	1.017	**1.000**

Computation Times - Regression

n	Lasso	SGD	SIHT	SADMM	OLSth	OFSA	OALa	OElnet	OMCP	RAVE
	ComputationTime (s) for Regression									
	$p = 1000, k = 100$, strong signal $\beta = 1$									
10^3	4.332	0.003	0.007	5.326	0.052	0.267	7.566	9.648	15.66	0.026
$3 \cdot 10^3$	26.91	0.010	0.019	15.73	0.051	0.267	2.972	7.113	10.21	0.076
10^4	47.32	0.032	0.065	51.80	0.051	0.266	2.404	5.885	7.123	0.246
	$p = 1000, k = 100$, weak signal $\beta = 0.1$									
10^3	3.989	0.003	0.006	5.387	0.051	0.266	7.258	7.706	16.30	0.027
$3 \cdot 10^3$	27.82	0.010	0.018	15.98	0.052	0.266	6.407	6.332	15.91	0.076
10^4	54.50	0.030	0.066	53.01	0.051	0.266	2.692	5.814	9.843	0.251
	$p = 1000, k = 100$, weak signal $\beta = 0.01$									
10^3	5.353	0.004	0.006	6.703	0.052	0.266	7.453	9.741	13.41	0.026
10^4	48.13	0.031	0.067	67.82	0.051	0.267	7.735	4.961	14.94	0.249
10^5	452.2	0.315	0.672	679.7	0.051	0.266	7.657	5.120	17.26	2.458
$3 \cdot 10^5$	1172	0.951	2.001	2044	0.051	0.267	5.977	3.749	13.10	7.326
10^6	-	3.158	6.651	-	0.051	0.267	3.602	1.726	7.866	24.36
	$p = 10000, k = 1000$, strong signal $\beta = 1$									
10^4	759.8	0.472	0.773	563.5	18.88	25.52	1129	1451	473.5	12.54
$3 \cdot 10^4$	2049	1.421	2.319	1687	18.81	26.07	484.0	1092	501.7	37.62
10^5	-	4.748	7.739	5633	19.00	26.01	415.7	983.9	462.5	124.8
	$p = 10000, k = 1000$, weak signal $\beta = 0.1$									
10^4	788.1	0.474	0.770	564.3	18.89	25.78	1284	1241	479.4	12.48
$3 \cdot 10^4$	1887	1.428	2.320	1689	18.92	25.96	696.5	859.1	434.2	37.41
10^5	-	4.747	7.747	5632	18.91	25.96	627.3	884.1	466.2	124.5
	$p = 10000, k = 1000$, weak signal $\beta = 0.01$									
10^4	827.4	0.473	0.773	564.6	18.91	25.95	1391	965.3	468.4	12.49
$3 \cdot 10^4$	1973	1.426	2.327	1693	18.89	26.12	1646	759.9	503.0	37.32
10^5	-	4.770	7.742	5662	18.81	25.99	1577	681.9	482.6	124.8
$3 \cdot 10^5$	-	14.29	23.21	16989	18.98	26.10	1521	741.6	481.4	373.0
10^6	-	47.72	77.40	-	19.02	26.11	1014	686.2	228.3	1242

Numerical Results - Classification

	Variable Detection Rate (%)								AUC							
	FOFS	SOFS	OPG	RDA	OFSA	OLSth	OLasso	OMCP	FOFS	SOFS	OPG	RDA	OFSA	OLSth	OLasso	OMCP
$p = 1000, k = 100$, strong signal $\beta = 1$																
10^4	10.64	10.19	10.46	10.97	38.89	30.30	34.70	**41.54**	0.995	0.992	0.992	0.990	0.995	0.990	**0.996**	**0.996**
3×10^4	10.64	9.95	10.42	10.34	**67.67**	59.32	56.18	67.52	0.994	0.992	0.992	0.989	**0.998**	0.996	0.997	**0.998**
10^5	10.64	9.95	10.43	11.08	**94.95**	93.21	86.90	94.77	0.994	0.992	0.992	0.990	**1.000**	**1.000**	0.999	**1.000**
$p = 1000, k = 100$, weak signal $\beta = 0.01$																
10^4	13.40	10.19	10.00	10.37	19.41	15.93	22.55	**23.81**	0.827	0.829	0.828	0.828	0.824	0.815	0.829	**0.830**
3×10^4	15.86	9.95	10.23	10.34	34.46	27.35	35.14	**37.70**	0.827	0.829	0.829	0.829	0.831	0.827	**0.832**	**0.832**
10^5	17.36	9.95	10.32	10.91	64.84	56.42	61.07	**64.95**	0.830	0.831	0.831	0.830	**0.834**	0.833	**0.834**	**0.834**
3×10^5	17.13	9.23	10.32	10.37	91.55	88.91	88.69	**91.58**	0.826	0.828	0.828	0.827	**0.833**	**0.833**	**0.833**	**0.833**
10^6	17.72	9.91	-	-	**99.97**	99.94	99.88	**99.97**	0.828	0.829	-	-	**0.834**	**0.834**	**0.834**	**0.834**

	Time (s)								
	FOFS	SOFS	OPG	RDA	OFSA	OLSth	OLasso	OMCP	RAVE
$p = 1000, k = 100$, strong signal $\beta = 1$									
10^4	0.001	0.001	0.490	0.848	0.005	0.001	0.080	0.160	0.247
3×10^4	0.003	0.004	1.471	2.210	0.005	0.001	0.083	0.158	0.742
10^5	0.010	0.015	4.900	6.118	0.005	0.001	0.079	0.159	2.478
$p = 1000, k = 100$, weak signal $\beta = 0.01$									
10^4	0.001	0.001	0.494	0.815	0.005	0.001	0.073	0.148	0.249
3×10^4	0.003	0.004	1.481	2.093	0.005	0.001	0.074	0.152	0.743
10^5	0.010	0.015	4.935	5.827	0.005	0.001	0.078	0.161	2.472
3×10^5	0.030	0.044	14.81	17.31	0.005	0.001	0.073	0.164	7.446
10^6	0.100	0.146	-	-	0.005	0.001	0.039	0.110	24.85

Regret Analysis

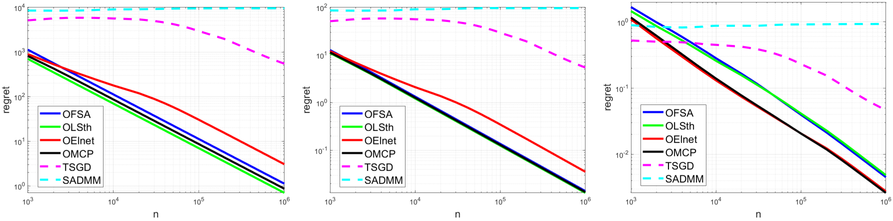

Figure: Regret vs n for online algorithms, averaged from 20 runs. Left: strong signal, middle: medium signal, right: weak signal

Real Data Analysis

Table: The average R^2 for regression and AUC for classification.

Dataset	n	p	OLSth	OFSA	Lasso	TSGD	SADMM
Regression data							
WIKIFace	53040	4096	**0.547**	0.545	0.503	0.400	0.487
Year Pred. MSD (nonlin.)	463715	4185	**0.303**	0.298	-	0	0
Year Prediction MSD	463715	90	**0.237**	**0.237**	**0.237**	0.157	0.183
	n	p	OLSth	OFSA	Lasso	FOFS	SOFS
Classification data							
Gisette	7000	5000	0.990	**0.997**	0.993	0.566	0.502
Dexter	600	20000	0.936	**0.971**	0.940	0.499	0.499

- Average of 20 random splits.
- For each method, multiple models are trained using various values of the tuning parameters and sparsity levels k.

Model Adaptation

- In online learning, the model that generates the data can change in time.
- We would like the estimated model to adapt as well.
- The running averages are updated using

$$\boldsymbol{\mu}_x^{(n+1)} = (1-\alpha_n)\boldsymbol{\mu}_x^{(n)} + \alpha_n \mathbf{x}_{n+1}$$

  with $\alpha_n = 1/(n+1)$.
- For adaptation, use larger α_n, e.g. $\alpha_n = 0.01$.
- Gives larger weight to recent observations

Model Adaptation

- Model update

$$\boldsymbol{\mu}_x^{(n+1)} = (1-\alpha_n)\boldsymbol{\mu}_x^{(n)} + \alpha_n \mathbf{x}_{n+1}$$

with $\alpha_n = 0.01$.

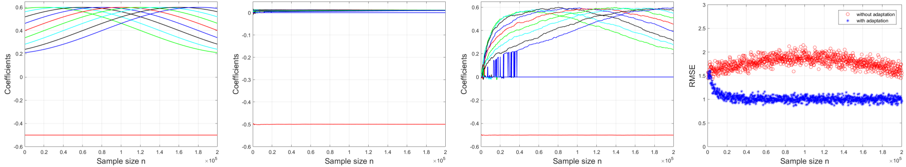

Figure: From left to right: true signal, parameters w/o adaptation, parameters w/ adaption, prediction RMSE.

Conclusion and Summary

- A framework based on running averages
- Data standardization and feature selection
- Online versions of many current feature selection methods
- Good performance in experiments w.r.t. other online or offline methods
- Advantages
  - Can recover the support of the true signal with high probability
  - Good convergence rate and low computation complexity
- Disadvantages
  - A very large p will run out of memory
- Possible application: federated learning

A. Barbu, Y. She, L. Ding, and G. Gramajo. Feature selection with annealing for computer vision and big data learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(2):272–286, 2017.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10(Dec): 2899–2934, 2009.

J. C. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari. Composite objective mirror descent. In *COLT*, pages 14–26, 2010.

J. Fan, W. Gong, C. J. Li, and Q. Sun. Statistical sparse online regression: A diffusion approximation perspective. In *AISTATS*, pages 1017–1026, 2018.

E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

J. Kiefer, J. Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3): 462–466, 1952.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10(Mar):777–801, 2009.

M. Neykov, J. S. Liu, and T. Cai. L1-regularized least squares for support recovery of high dimensional single index models with gaussian designs. *Journal of Machine Learning Research*, 17(87):1–37, 2016.

H. Ouyang, N. He, L. Tran, and A. Gray. Stochastic alternating direction method of multipliers. In *ICML*, pages 80–88, 2013.

H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

S. Shalev-Shwartz and A. Tewari. Stochastic methods for l1-regularized loss minimization. *Journal of Machine Learning Research*, 12(Jun): 1865–1892, 2011.

T. Suzuki. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *ICML*, pages 392–400, 2013.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Y. Wu, S. C. Hoi, T. Mei, and N. Yu. Large-scale online feature selection for ultra-high dimensional sparse data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(4):48, 2017.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct): 2543–2596, 2010.

C.-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics*, pages 894–942, 2010.

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.